An Index-first Addressing Scheme for Multi-level Caches

Hemant Salwan
Department of Information technology,
Institute of Information Technology and Management, New Delhi, India
salwan.hemant@gmail.com

***Abstract*—Performance is one of the leading factors in the current processor designs. A processor's efficiency highly depends upon the organization of its cache. In the multi-level caches, the existing scheme of addressing the cache memory produces a significant fraction of address conflicts at the lower level cache. This increases the global miss rate which diminishes the cache and processor performance. This problem becomes more severe in multi-processing environment because of conflicts of inter-process interference. An approach that performs the cache addressing efficiently in multi-level caches is proposed in this paper. The experimental results show that this scheme significantly decreases the global miss rate of the caches and the cycles per instruction (CPI) performance metric of the processor.**

***Index Terms*—cache memory, cache addressing, conflict misses, multi-level caches**

I. Introduction

Cache memory addressing scheme [1] partitions the physical address into three fields: tag, index and offset. The index field is used to select one of the cache sets. The tag field is compared against it for a hit. Figure 1 shows how an address is divided into three fields. The block offset selects the desired word from the block.

Figure 1. Traditional address partitioning

As per to this traditional scheme the contiguous blocks in memory are mapped to contiguous blocks of the cache. Although this scheme uniformly distributes the memory block addresses across the cache block frames, but it suffers with two major limitations [2] [3] [4]. First, multiple memory blocks map to the same set of block frames at all the cache levels which potentially produces global conflict misses in the caches. Second, due to generally much larger high level cache multiple blocks of higher level cache map to a single frame of lower level cache which causes conflict misses at the lower level. In Figure 4 the types of block addressing conflicts found in the traditional scheme are illustrated. These limitations arise specifically in two workload environments: i) a single running process whose memory access pattern is scattered across the memory pages instead of being localized to a region. This scenario is specially encountered when the size of primary cache is very small, and ii) in the multi-processing environment where the caches are not large enough to accommodate the working sets of all the running processes without conflict. This scenario is most significant in the current multi-processor systems which allow a high degree of multi processing causing even large caches to crunch under its pressure.

Previous studies have explored these drawbacks and proposed the solutions. Non–temporal streaming [5] has reduced the cache conflict misses by predicting the temporal behavior of a block. Alternative cache hashing functions were used in [6] to eliminate the conflicts misses. Cache Miss Look aside [7] used a hardware buffer to reduce conflict misses. Compile-time transformations [8] [9] were also used to eliminate conflict misses. Unfortunately, all of these schemes either require a specialized hardware or are not dynamic in nature.

An efficient scheme of addressing the cache is proposed in this paper which overcomes the above mentioned limitations in the above said workload environments. Moreover, it does not require any additional hardware on the processor chip. Section II describes the index-first cache addressing scheme in detail and also discuss its comparison with the traditional scheme. Section III describes the experimental methodology and the results achieved using the new scheme. Finally, Section IV concludes the paper.

II. Index-First Addressing

In this proposed cache addressing scheme, we interchange the tag and index fields of the traditional way to make it called as index-first addressing. This scheme thus uses the high order bits to index into the cache and lower order bits as the tag to compare against the selected cache block frame for a hit. The address fields are as shown in Figure 2.

Figure 2.Index-first address partitioning

The following relations will hold true in the index-first addressing scheme:

$$(\text{No. of cache blocks}) = (\text{No. of regions}) = 2^{\text{index}}$$
$$(\text{Region Size}) = 2^{\text{tag+offset}}$$
$$(\text{Block size}) = 2^{\text{offset}}$$

By this way of addressing the memory, the whole memory can be seen as composed of a fixed number of regions and each region is composed of a fixed number of blocks. It is illustrated in Figure 3 how this scheme maps in a solo cache.

DOI: 03.LSCS.2013.2.40

It should be clear from the figure that all the blocks of a region maps to a single block frame in the cache and no two blocks of different regions map to the same block frame.

This logical division of memory into regions is similar to what is called as paging in memory management. The region size should be comparable to a page but it is preferable that the region size be smaller than the page size in order that the program memory references remain scattered across the regions as much as possible.

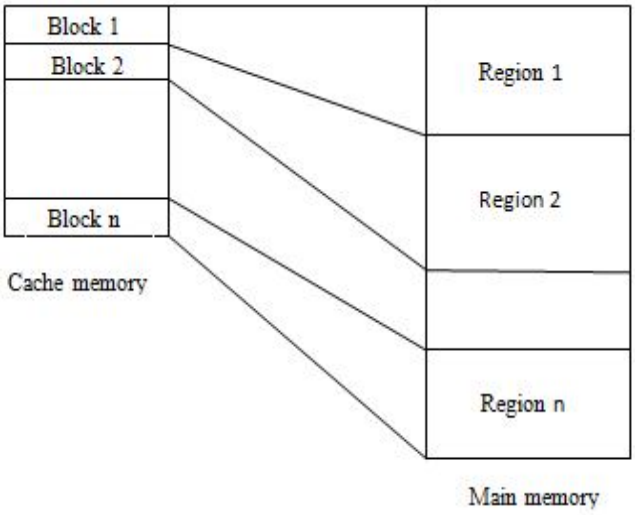


Figure 3. Address mapping as per the index-first scheme

block 2 to frame 2 in traditional and frame 1 in index-first. It must be noted that all memory accesses will belong to the category of one of these four mappings. Table I enumerates the all the possible block mapping conflicts that may occur in the two addressing schemes. The table shows the caches that conflict for a particular mapping pair and also the number of memory blocks (called degree) involved in such a conflict. A degree of one means no conflict.

TABLE I. MAPPING CONFLICTS FOR THE TWO SCHEMES

Mapping pair in conflict	Degree of conflict	
	Traditional	Index-first
(1,2)	1	M/m
(1,3)	M/m	M/m^2
(1,4)	M/n	M/mn

The above analysis shows the proposed scheme produces the following effects on the caches:

a. The mapping pair (1, 2) produces the single conflicts in the L1 cache in the index-first scheme. This effect can be lowered by keeping the M/m value lower. This can be done by using a larger sized L1 cache.
b. The mapping pair (1, 3) causes a large no. of single conflicts in the L1 cache in traditional scheme. Or conversely, the index-first scheme reduces such single conflicts in the L1 cache by a factor of 1/m. This effect can be made higher by increasing the value of 1/m. This can be done by using a comparatively smaller L1 cache.
c. The mapping pair (1, 4) produces a large no. of global

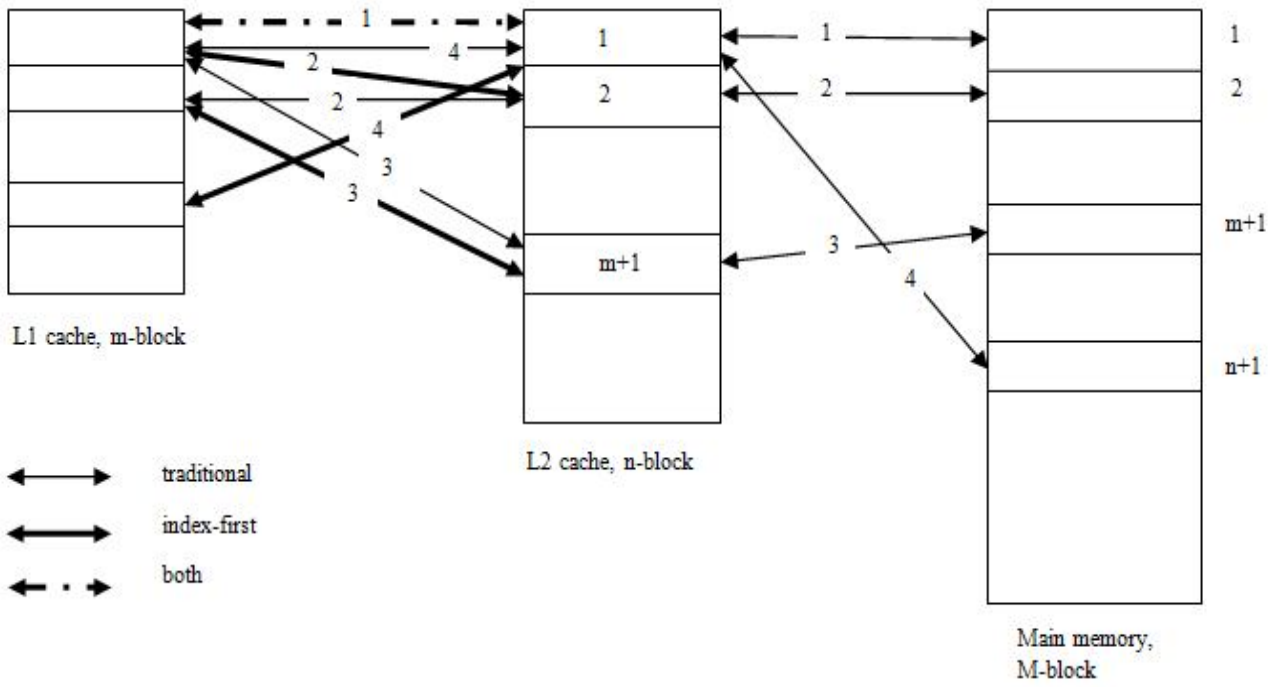
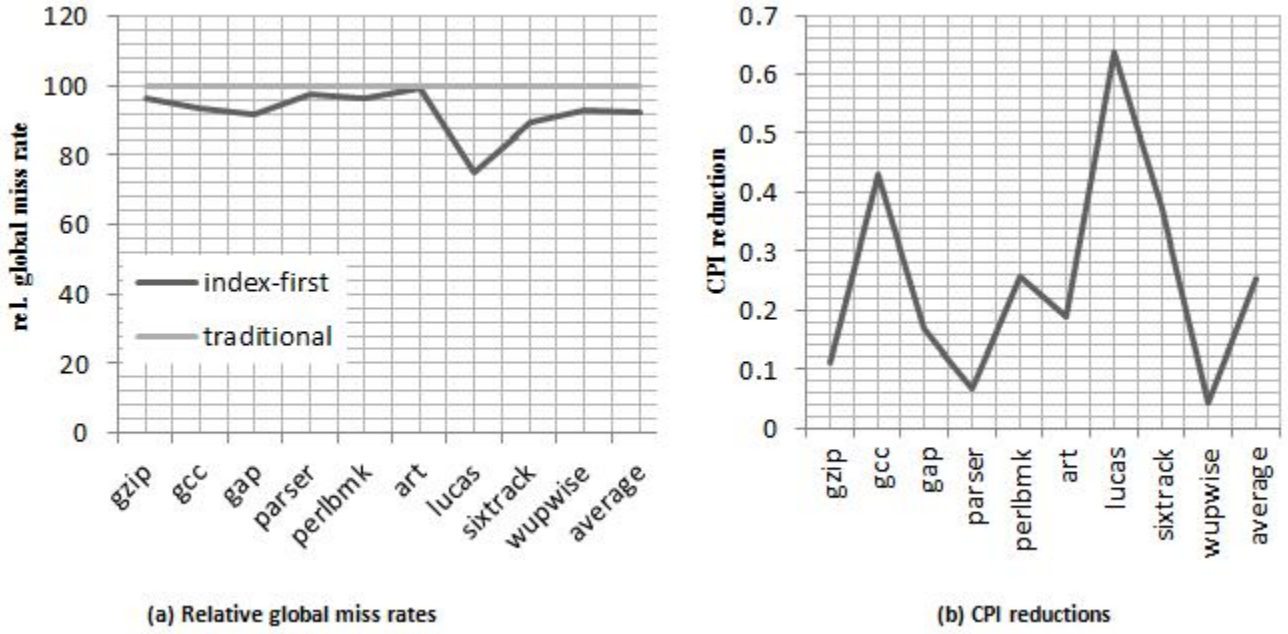


Figure 4. Addressing conflicts using index-first vs. traditional scheme. Block size is uniform at all the levels

The addressing conflicts that occur in the caches when using index-first scheme versus traditional scheme are as shown in Fig. 4. Note that it is only the L1$ at which the block mappings of two schemes differ. Each numeric label on the mappings denotes a particular type of memory block access. For example mapping "1" shows that both schemes map memory block 1 to frame 1 of the L1$ whereas mapping "2" maps

DOI: 03.LSCS.2013.2.40



(a) Relative global miss rates

(b) CPI reductions

Figure 5. Single-trace simulations of a set of benchmarks

conflicts in the traditional scheme whereas many of these are eliminated in the index-first scheme. This will reduce the global miss rate in the index-first scheme again by a factor of 1/m. This effect can also be made larger by using a smaller sized L1 cache.

The effects (a), (b), and (c) depict the reason of applying the index-first addressing scheme only at the lower level instead of higher levels. The important observation from the effect (c) is that index-first scheme services many requests at L1 level which are serviced by memory in traditional scheme. This will significantly improves the CPU performance because of a large difference between L1 cache access time and memory access time. Another important observation is that the (1, 2) conflict arise out of the same page, (1, 3) conflicts may or may not arise out of the same page, and the (1, 4) conflicts only arise out of the different pages. This implies that in multi-processing systems index-first scheme will give far better CPU performance gain as compared to single processing environment because effect (b) and (c) will be dominating in the multi-processing systems.

III. Experiment And Results

Our goal is to evaluate the effect of index-first addressing scheme on the cache and CPU performance as specified in the previous section. To do this, global cache miss rate and instructions per cycle (IPC) were used as the performance metrics. The cache hierarchy system that was modeled is detailed in Table II.

A. Simulation environment

The Dinero IV cache simulator [10] was used to run the traces of various benchmarks. The simulator was modified to include the index-first addressing scheme. The simulator was extended to simulate the multi-processing environment by running 2- and 4- traces simultaneously. In the multi-trace simulation the switching period of 100 instructions was used.

Table II. Cache Hierarchy Model Parameters

Parameters	L1 Cache	L2 Cache
Cache size (KB)	4	64
Block size (B)	64	64
Associativity	1	1
Type	Unified	Unified
Miss penalty (Cycles)	20	200

B. Benchmarks

The SBC traces [11] of SPEC CPU2000 benchmarks [12] suite were used in the trace simulation. However not all the benchmarks from the suite could be used in the simulation, so a set of nine benchmarks (five integers and four floating point) were selected from the benchmark suite for simulation purposes. Each trace included the memory references of first two billion of instruction execution.

C. Results

The results of the simulations of running singleton traces are shown in Figure 5. Fig. 5 (a) shows the global miss rates of the index-first scheme relative to the traditional scheme and Fig. 5 (b) shows the corresponding CPI reductions achieved in the index-first scheme. In index-first scheme, average global miss rate decreased to around 92% of the original and lowest to 75% of traditional for lucas. In the index-first scheme, the average reduction in CPI is over 0.25 and a maximum CPI reduction of 0.63 for lucas.

The relative global miss rates and CPI reductions of the index-first scheme of running 2-trace and 4-trace simulations are shown in Figure 6. Figure 6 (a) shows that the average global miss rate decreased to around 89% of traditional for 2-trace simulations and around 85% of traditional for 4-trace

DOI: 03.LSCS.2013.2.40

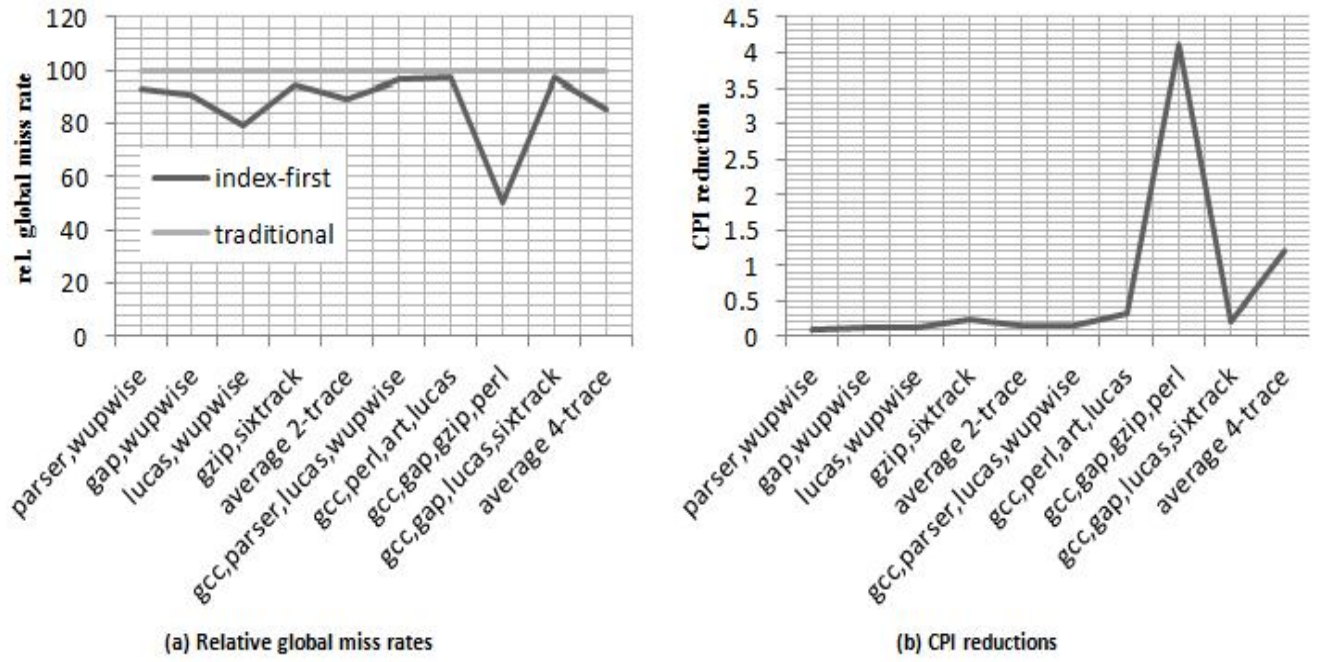


Figure 6. Multi-trace simulations of a set of 2-trace and 4-trace benchmarks

simulations. It maximum decreased to 50% of traditional for (gcc,gap,gzip,perl) and to 79% of traditional for (lucas,wupwise). Fig. 6 (b) shows that the average CPI reduction of index-first scheme is over 0.14 for 2-trace and around 1.2 for 4-trace simulations.

IV. Conclusions

An index-first cache addressing scheme has been presented which is an efficient scheme for addressing the cache memories. Using this scheme the first portion of the memory address is used for indexing the L1 cache instead of second as used in the traditional scheme. This scheme works by eliminating several types of block address conflicts in a multi-level cache system. This scheme can be implemented without any additional hardware. The results show that this scheme outperforms the traditional scheme across all the benchmarks in both global miss rates and CPI measures. Moreover, the performance improvements were found to be many times more in multi-processing environments.

References

[1] Alan Jay Smith. "Cache memories". ACM Computing Surveys, pp. 473-530, Sept. 1982.

[2] Monica S. Lam, Edward E. Rothberg, and Michael E. Wolf. "The cache performance and optimizations of blocked algorithms". ACM SIGARCH Computer Architecture News, pp. 63-74, 1991.

[3] R. C. Murphy and Peter M. Kogge. "On the memory access patterns of supercomputer applications: Benchmark selection and its implications". IEEE Transactions on Computers, 7(56), July 2007.

[4] S. Przybylski, M. Horowitz, and J. Hennessy. "Characteristics of performance-optimal multi-level cache hierarchies". In Proc. of the 16th Annual Int. Symposium on Computer Architecture, pp. 114-121, June 1989.

[5] J. A. Rivers. "Reducing conflicts in direct-mapped caches with a temporality-based design". In Proc. of the Int. Conference on Parallel Processing, pp. 154-163, August 1996.

[6] M. Kharbutli, K. Irwin, Y. Solihin, and J. Lee. "Using prime numbers for cache indexing to eliminate conflict misses". IEEE Proc. on Software, pp. 288-299, Feb. 2004.

[7] Brian N. Bershad, Dennis Lee, Theodore H. Romer, and J. Bradley Chen. "Avoiding conflict misses dynamically in large direct-mapped caches". In Proc. of the 6th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, pp. 158-170, 1994.

[8] Gabriel Rivera and Chau-Wen Tseng. "Data transformations for eliminating conflict misses". In Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation, pp 38-49, 1998.

[9] Gabriel Rivera and Chau-Weng Tseng. "Eliminating conflict misses for high performance architectures". In Proc. of the 12th Int. Conf. on Supercomputing, pp. 353-360, 1998.

[10] Mark D. Hill. Dinero IV Trace-driven uniprocessor cache simulator. "pages.cs.wisc.edu/~markhill/DineroIV".

[11] A. Milenkovic, M. Milenkovic. "Exploiting streams in instruction and data address trace compression". In Proc. of the IEEE 6th Annual Workshop on Workload Characterization, October 2003.

[12] J. L. Henning. "SPEC CPU2000: Measuring CPU performance in the new millennium". IEEE Computer, July 2000.

DOI: 03.LSCS.2013.2.40